Cache Memory

Reading: Section 4.2 & Section 4.3 on pp. 108-127

The goal of cache memory is to yield a memory speed of the fastest memories available while at the same time providing a much larger memory at the cheapest possible price.

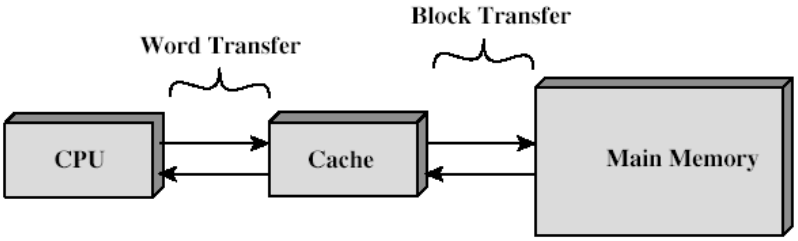


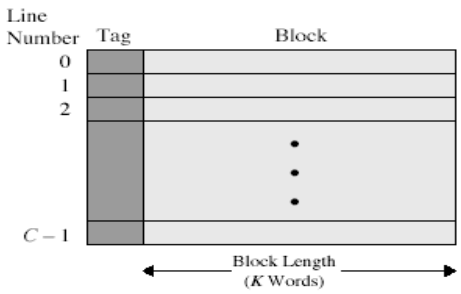
Figure 4.3 Cache and Main Memory

The above figure illustrates the following:

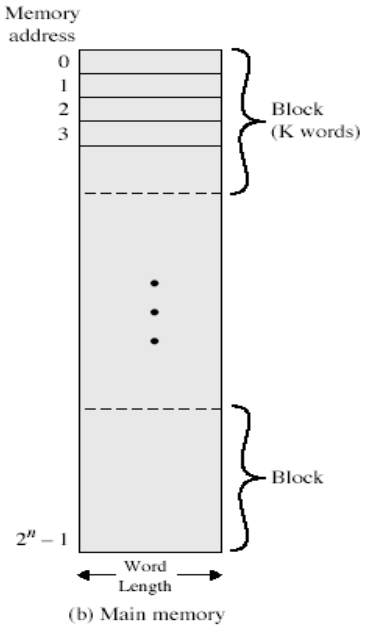
1) The processor attempts to read a word of memory from the cache.
2) If the word exists in the cache, the word is transferred to the CPU.
3) If the word does not exist, a block of memory is read from main memory and transferred into the cache. The block transferred contains the word originally attempted to be read.

P1: Assume that a L1 cache contains 1000 words and an access time of .01 μs and a L2 cache contains 100,000 words and has an access time of .1 μs. If 95% of the memory accesses are found in the L1 cache, what is the average access time to access a word from the two levels of cache?

A cache/main memory structure looks like the following:



(a) Cache



(b) Main memory

Things to note:

1) A main memory has 2^n addressable memory locations.
2) Each word has a unique n-bit address.
3) A block of K words is mapped to a line in the cache.

Q1: How many blocks of information exist in main memory?

4) The number of words in a cache line is called the line size.

Q2: What is the line size in the above picture?

Q3: If M is the number of main memory blocks and C is the number of cache lines, would you say that C is a little smaller than M or a lot smaller than M? Why?

5) Because there are more blocks than cache lines, a tag field is used to help identify which block of main memory is in the cache. This will become more clear when we get into specific cache mapping strategies.

Let's go through a cache read operation where RA denotes a read address.

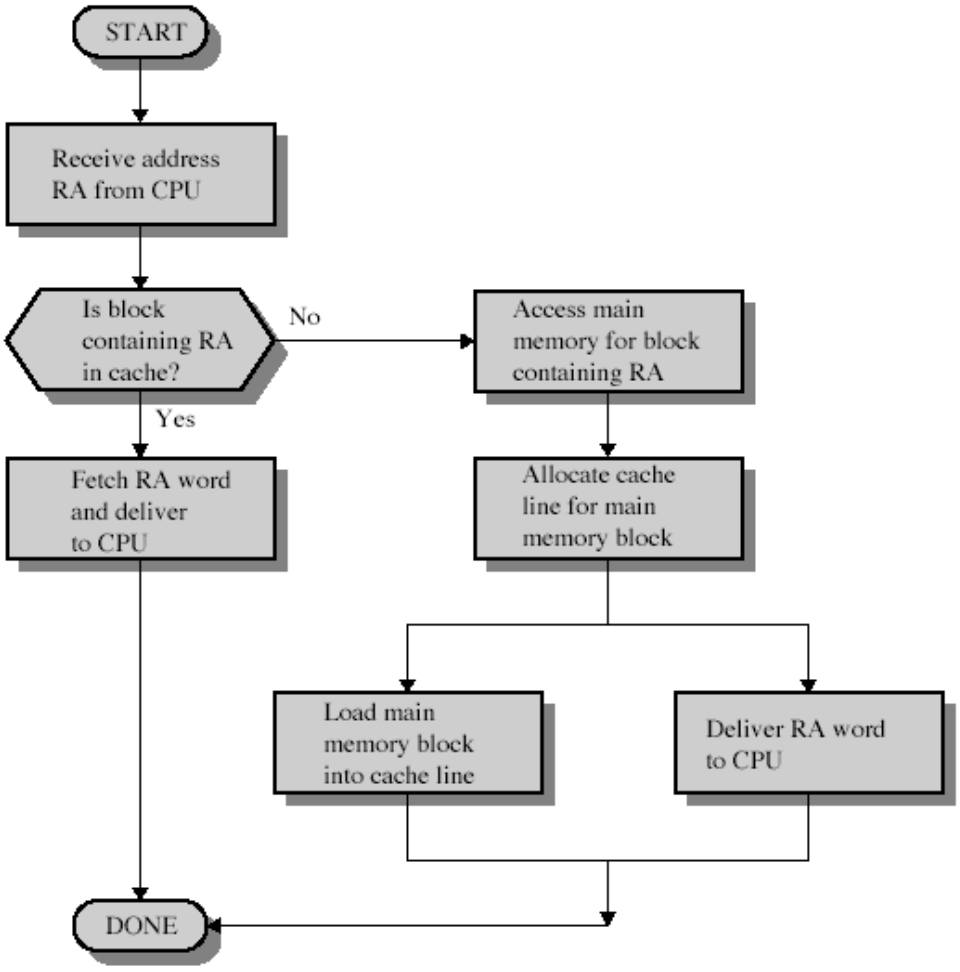
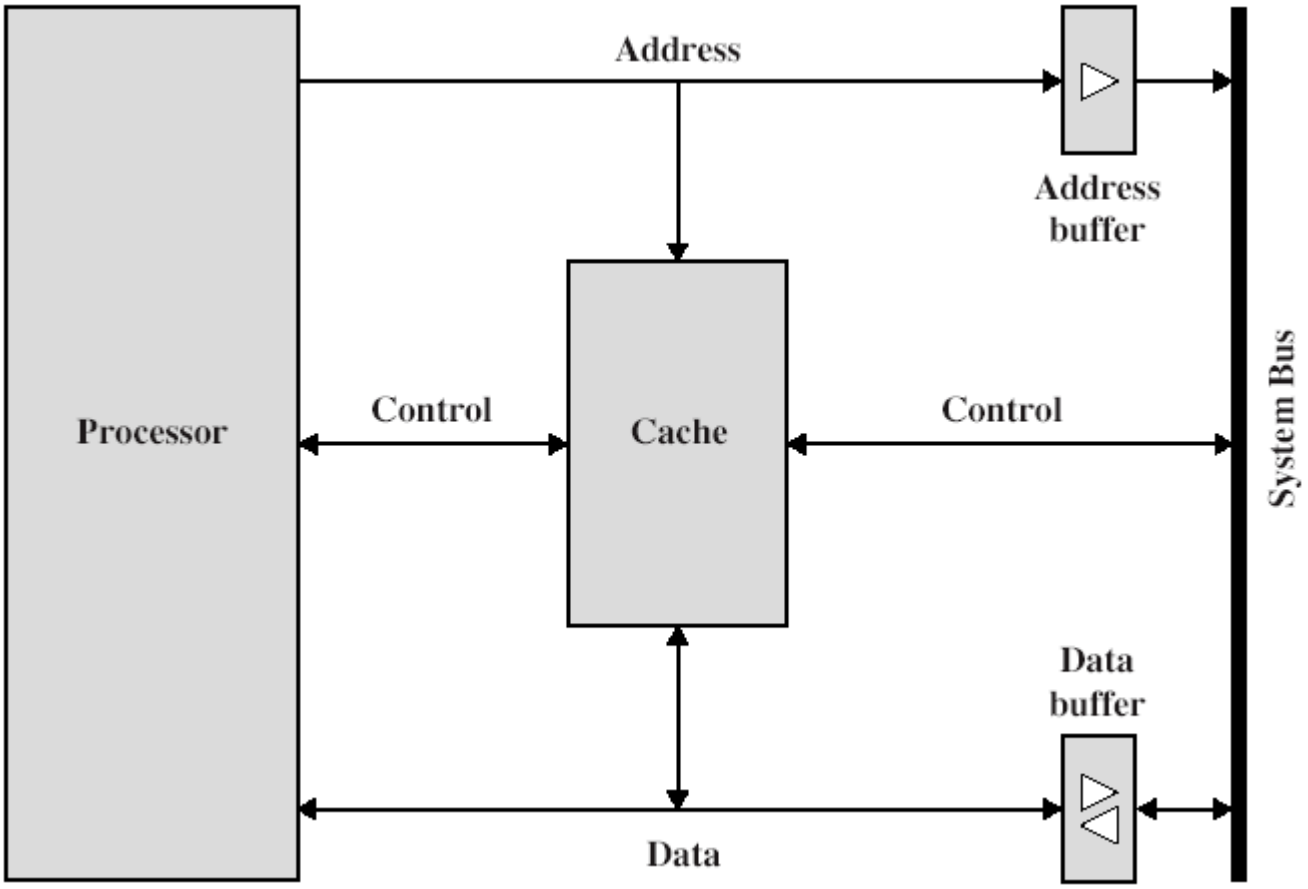


Figure 4.5 Cache Read Operation



Cache Size

In a perfect world, the size of the cache is small enough so that the cost per bit is close to that of main memory, but large enough so that the average access time is close to that of the cache.

Other issues include:

a) The larger the cache, the more gates needed to access the cache, thus making a larger cache slower than a smaller cache.

b) Much of the time, the available chip and board area limit the size of the cache.

Let's take a look at some typical cache sizes for various processors over time.

Table 4.3 Cache Sizes of Some Processors

Processor	Type	Year of Introduction	L1 cache[a]	L2 cache	L3 cache
IBM 360/85	Mainframe	1968	16 to 32 KB	—	—
PDP-11/70	Minicomputer	1975	1 KB	—	—
VAX 11/780	Minicomputer	1978	16 KB	—	—
IBM 3033	Mainframe	1978	64 KB	—	—
IBM 3090	Mainframe	1985	128 to 256 KB	—	—
Intel 80486	PC	1989	8 KB	—	—
Pentium	PC	1993	8 KB/8 KB	256 to 512 KB	—
PowerPC 601	PC	1993	32 KB	—	—
PowerPC 620	PC	1996	32 KB/32 KB	—	—
PowerPC G4	PC/server	1999	32 KB/32 KB	256 KB to 1 MB	2 MB
IBM S/390 G4	Mainframe	1997	32 KB	256 KB	2 MB
IBM S/390 G6	Mainframe	1999	256 KB	8 MB	—
Pentium 4	PC/server	2000	8 KB/8 KB	256 KB	—
IBM SP	High-end server/ supercomputer	2000	64 KB/32 KB	8 MB	—
CRAY MTA[b]	Supercomputer	2000	8 KB	2 MB	—
Itanium	PC/server	2001	16 KB/16 KB	96 KB	4 MB
SGI Origin 2001	High-end server	2001	32 KB/32 KB	4 MB	—
Itanium 2	PC/server	2002	32 KB	256 KB	6 MB
IBM POWER5	High-end server	2003	64 KB	1.9 MB	36 MB
CRAY XD-1	Supercomputer	2004	64 KB/64 KB	1MB	—

[a] Two values seperated by a slash refer to instruction and data caches
[b] Both caches are instruction only; no data caches

(1) Direct Mapping

Direct mapping is the simplest of the cache mapping schemes. Direct mapping is expressed by the following formula:

i = j modulo m where

i = cache line number
j = main memory block number
m = number of lines in the cache

This mapping strategy can be implemented using the main memory address as follows:

a) The least significant w bits identify a unique byte within the block of main memory.

b) The remaining s bits represent one of 2^s blocks of main memory.

c) The cache uses these s-bits as a tag of s-r bits (most significant portion) and a line of r bits where 2^r cache lines exist.

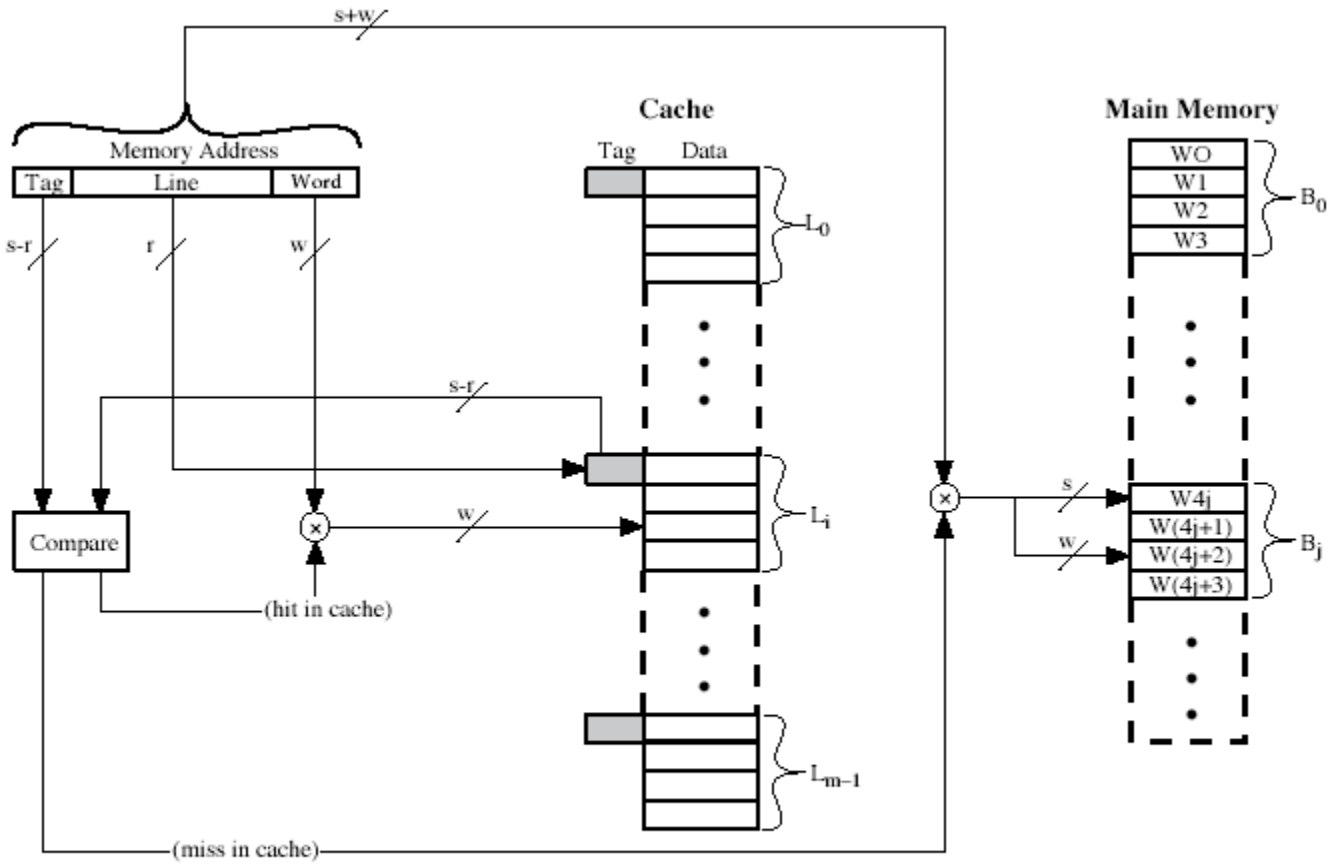


Figure 4.7 Direct-Mapping Cache Organization [HWAN93]

To summarize:

Address Length is (s + w) bits

Number of addressable units is 2^{s+w} bytes

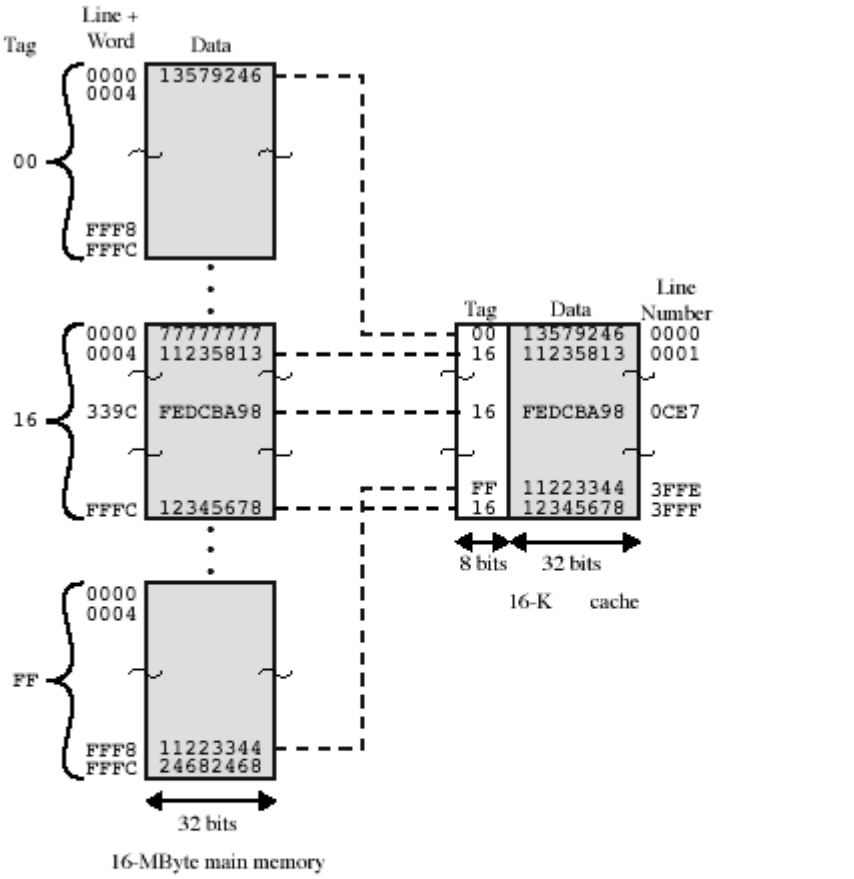
Block size = line size = 2^w bytes

Number of blocks in main memory is $\frac{2^{s+w}}{2^w} = 2^s$

Number of cache lines is 2^r

Tag size is (s-r) bits

Let's go through the following example:



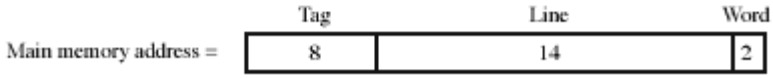


Figure 4.8 Direct Mapping Example